(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 105224449 B
(45)授权公告日 2017.12.12

(21)申请号 201410294770.1

(22)申请日 2014.06.26

(65)同一申请的已公布的文献号
申请公布号 CN 105224449 A

(43)申请公布日 2016.01.06

(73)专利权人 富士通株式会社
地址 日本神奈川县

(72)发明人 张沈斌 孙俊 张军 邹纲 周恩策

(74)专利代理机构 北京集佳知识产权代理有限公司 11227
代理人 朱胜 江河清

(51)Int.Cl.
G06F 11/36(2006.01)

(56)对比文件
CN 103885873 A,2014.06.25,
CN 101394646 A,2009.03.25,
US 2007277158 A1,2007.11.29,
CN 102693183 A,2012.09.26,
谢红霞 等.基于Android的自动化测试的设计与实现.《计算机时代》.2012,(第2期),
公磊 等.基于Android的移动终端应用程序开发与研究.《计算机与现代化》.2008,(第8期),

审查员 王仕超

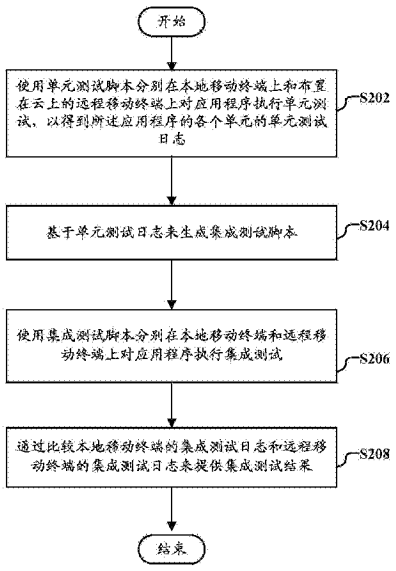
权利要求书2页 说明书13页 附图10页

(54)发明名称

移动终端上的应用程序的测试方法和装置

(57)摘要

本发明涉及一种移动终端上的应用程序的测试方法和装置，该方法包括：使用单元测试脚本分别在本地移动终端上和布置在云上的远程移动终端上对应用程序执行单元测试，以得到所述应用程序的各个单元的单元测试日志；基于所述单元测试日志来生成集成测试脚本；使用所述集成测试脚本分别在所述本地移动终端和所述远程移动终端上对所述应用程序执行集成测试；以及通过比较本地移动终端的集成测试日志和远程移动终端的集成测试日志来提供集成测试结果。根据本发明，可以不需要购买任何真机进行设备多样性的差异测试，从而节省成本；通过分析与候选设备相似的测试设备的集成测试结果，判断候选设备是否适配于应用程序，可以给测试者推荐所有适配手机类型。



CN 105224449 B

1.一种移动终端上的应用程序的测试方法,包括:

使用单元测试脚本分别在本地移动终端上和布置在云上的远程移动终端上对应用程序执行单元测试,以得到所述应用程序的各个单元的单元测试日志;

基于所述单元测试日志来生成集成测试脚本;

使用所述集成测试脚本分别在所述本地移动终端和所述远程移动终端上对所述应用程序执行集成测试;以及

通过比较本地移动终端的集成测试日志和远程移动终端的集成测试日志来提供集成测试结果,

其中,基于所述单元测试日志来生成集成测试脚本包括:

通过分析所述单元测试日志来计算各个单元的单元错误率,所述单元错误率是该单元的单元测试的错误数与全部单元的单元测试的错误的总数的比率;

至少基于各个单元的所述单元错误率来估算各个单元的单元错误概率,所述单元错误概率是每个单元的错误发生的概率;

基于各个单元的所述单元错误概率生成集成测试路径;以及

利用所述集成测试路径来生成所述集成测试脚本。

2.根据权利要求1所述的方法,还包括:利用所述集成测试结果,进行候选移动终端与所述应用程序的适配性测试。

3.根据权利要求1所述的方法,其中,所述单元测试的错误数通过如下公式计算:

$$\text{单元测试的错误数} = \sum_{i=1}^{N_1} \text{上下文无关错误}_i * \gamma^{i-1} + \sum_{i=1}^{N_2} \text{上下文相关关错误}_i * \gamma^{i-1}$$

其中,N_1是本地测试的次数,N_2是远程测试的次数,γ是折扣率。

4.根据权利要求1所述的方法,其中,至少基于各个单元的单元错误率来计算各个单元的单元错误概率进一步包括:

基于各个单元的单元错误率和错误预测概率来加权计算各个单元的单元错误概率,所述错误预测概率是对单元错误概率的预测。

5.根据权利要求1所述的方法,其中,基于各个单元的单元错误概率生成集成测试路径包括:

将所述应用程序的源代码转换为所有单元的图结构,优先选择单元错误概率最大的单元遍历所述图结构来生成集成测试路径。

6.根据权利要求1所述的方法,其中,利用所述集成测试路径来生成集成测试脚本包括:

基于模型－视图－控制器模式,通过循环探测生成满足所述集成测试路径上的单元跳转条件的测试数据,根据所述测试数据和所述集成测试路径来生成集成测试脚本。

7.根据权利要求1所述的方法,其中,将所述应用程序的一个视图页面作为一个单元。

8.根据权利要求2所述的方法,其中,利用所述集成测试结果,进行候选移动终端与所述应用程序的适配性测试包括:

计算所述候选移动终端与执行所述集成测试的测试移动终端之间的相似度;

选择与所述候选移动终端的相似度大于第一预定阈值的测试移动终端的集成测试结

果来计算所述候选移动终端与所述应用程序的适配概率;以及

选择与所述应用程序的所述适配概率大于第二预定阈值的候选移动终端作为适配终端。

9.一种移动终端上的应用程序的测试装置,包括:

单元测试单元,被配置为使用单元测试脚本分别在本地移动终端上和布置在云上的远程移动终端上对应用程序执行单元测试,以得到所述应用程序的各个单元的单元测试日志;

集成测试脚本生成单元,被配置为基于所述单元测试日志来生成集成测试脚本;

集成测试单元,被配置为使用所述集成测试脚本分别在所述本地移动终端和所述远程移动终端上对所述应用程序执行集成测试;以及

测试结果提供单元,被配置为通过比较本地移动终端的集成测试日志和远程移动终端的集成测试日志来提供集成测试结果,其中,所述集成测试脚本生成单元进一步被配置为:

通过分析所述单元测试日志来计算各个单元的单元错误率,所述单元错误率是该单元的单元测试的错误数与全部单元的单元测试的错误的总数的比率;

至少基于各个单元的所述单元错误率来估算各个单元的单元错误概率,所述单元错误概率是每个单元的错误发生的概率;

基于各个单元的所述单元错误概率生成集成测试路径;以及

利用所述集成测试路径来生成所述集成测试脚本。

移动终端上的应用程序的测试方法和装置

技术领域

[0001] 本发明涉及移动应用测试领域，更具体地涉及一种移动终端上的应用程序的测试方法和装置。

背景技术

[0002] 如今，移动互联网随着智能手机和3G/4G网络的普及，越来越繁荣。人们逐渐开始使用智能手机，企业和个人开发的app(移动应用程序)每天都以成千上万的数量登录各大app商城。然而，由于缺乏有效的app测试方法，app的质量并不是很高，经常无法满足移动用户的需求。

[0003] 传统的测试方法中，在开发完app之后，使用模拟器或者手机真机对app进行测试。然而，app测试与传统的PC(个人计算机)软件测试不同，传统的本地测试方法无法满足要求。由于设备差异性的存在，若要对app进行充分地测试，必须购买相当数量的手机，这种方法使得测试成本相当高昂。此外，传统的测试方法也无法自动化地生成测试脚本，例如，monkeyrunner和robotium等都无法智能地生成测试脚本。

[0004] 目前的手机app测试方法中，当测试者进行app测试时，由于可能缺乏足够数量的手机，从而无法完成设备多样性测试，而且测试者必须手动编写测试脚本。由于缺乏大数据的分析，测试者编写的测试脚本的覆盖率往往也不是很高。

[0005] 因此，需要一种能够对移动终端上的应用程序高效进行测试的方法和装置。

发明内容

[0006] 在下文中给出关于本发明的简要概述，以便提供关于本发明的某些方面的基本理解。应当理解，这个概述并不是关于本发明的穷举性概述。它并不是意图确定本发明的关键或重要部分，也不是意图限定本发明的范围。其目的仅仅是以简化的形式给出某些概念，以此作为稍后论述的更详细描述的前序。

[0007] 本发明的一个主要目的在于，提供一种移动终端上的应用程序的测试方法，包括：使用单元测试脚本分别在本地移动终端上和布置在云上的远程移动终端上对应用程序执行单元测试，以得到所述应用程序的各个单元的单元测试日志；基于所述单元测试日志来生成集成测试脚本；使用所述集成测试脚本分别在所述本地移动终端和所述远程移动终端上对所述应用程序执行集成测试；以及通过比较本地移动终端的集成测试日志和远程移动终端的集成测试日志来提供集成测试结果。

[0008] 根据本发明的一个方面，提供了一种移动终端上的应用程序的测试装置，包括：单元测试单元，被配置为使用单元测试脚本分别在本地移动终端上和布置在云上的远程移动终端上对应用程序执行单元测试，以得到所述应用程序的各个单元的单元测试日志；集成测试脚本生成单元，被配置为基于所述单元测试日志来生成集成测试脚本；集成测试单元，被配置为使用所述集成测试脚本分别在所述本地移动终端和所述远程移动终端上对所述应用程序执行集成测试；以及测试结果提供单元，被配置为通过比较本地移动终端的集成

测试日志和远程移动终端的集成测试日志来提供集成测试结果。

[0009] 另外，本发明的实施例还提供了用于实现上述方法的计算机程序。

[0010] 此外，本发明的实施例还提供了至少计算机可读介质形式的计算机程序产品，其上记录有用于实现上述方法的计算机程序代码。

[0011] 通过以下结合附图对本发明的最佳实施例的详细说明，本发明的这些以及其他优点将更加明显。

附图说明

[0012] 参照下面结合附图对本发明实施例的说明，会更加容易地理解本发明的以上和其它目的、特点和优点。附图中的部件只是为了示出本发明的原理。在附图中，相同的或类似的技术特征或部件将采用相同或类似的附图标记来表示。

[0013] 图1示出了根据本发明的一个实施例的对移动终端上的应用程序进行测试以及进行适配手机推荐的框架的总体系统结构图；

[0014] 图2示出了根据本发明的一个实施例的对移动终端上的应用程序进行测试的方法的流程图；

[0015] 图3示出了测试者对app进行单元测试的示意流程图；

[0016] 图4是示出图2中的步骤S204的一种示例性过程的流程图；

[0017] 图5是示出集成测试路径生成的一个示例过程的示意图；

[0018] 图6是示出基于MVC的集成测试运行的一个示例的示意图；

[0019] 图7是示出了基于MVC进行集成测试的示意图；

[0020] 图8是示出了对适配手机类型进行推荐的一个示例过程的示意图；

[0021] 图9是示出根据本发明的一个实施例的移动终端上的应用程序的测试装置900的示例性配置的框图；

[0022] 图10示出根据本发明的另一个实施例的移动终端上的应用程序的测试装置900'的另一种示例性配置的框图；

[0023] 图11是示出图9中的集成测试脚本生成单元904的一种示例性配置的框图；

[0024] 图12是示出图10中的移动终端适配性测试单元910的一种示例性配置的框图；以及

[0025] 图13是示出可以用于实施本发明的对移动终端上的应用程序进行测试的方法和装置的计算设备的示例性结构图。

具体实施方式

[0026] 下面参照附图来说明本发明的实施例。在本发明的一个附图或一种实施方式中描述的元素和特征可以与一个或更多个其它附图或实施方式中示出的元素和特征相结合。应当注意，为了清楚的目的，附图和说明中省略了与本发明无关的、本领域普通技术人员已知的部件和处理的表示和描述。

[0027] 在本发明中，提出了一种基于自动化集成测试和手机适配的移动云测试框架。使用该测试框架，测试者可以使用云平台上的大量手机对app(移动应用程序)进行测试。测试框架提供了单元测试和集成测试，并且基于单元测试历史数据，可以帮助测试者自动化地

生成集成测试脚本。本框架的另外一个特点是提供了适配手机推荐模块，可以根据集成测试的结果，通过计算相似设备的测试成功概率，来推荐出所有的适用于该app的手机类型。图1示出了根据本发明的一个实施例的对移动终端上的应用程序进行测试以及进行适配手机推荐的框架的总体系统结构图。

[0028] 在下文中，以android系统上的app测试为例进行阐述。在下文中，以手机作为移动终端的一个实施例进行说明，本领域技术人员可以理解，根据本发明的一个实施例的移动终端上的应用程序的测试方法和装置可以应用于不限于手机的其他移动终端。在下文中，手机有时也描述为设备、Device等。

[0029] 下面结合附图详细说明根据本发明的一个实施例的对移动终端上的应用程序进行测试的方法。

[0030] 图2示出了根据本发明的一个实施例的对移动终端上的应用程序进行测试的方法的流程图。

[0031] 首先，在步骤S202中，使用单元测试脚本分别在本地移动终端上和布置在云上的远程移动终端上对应用程序执行单元测试，以得到所述应用程序的各个单元的单元测试日志。

[0032] 图3示出了测试者对app进行单元测试的示意流程图。

[0033] Android app的UI(用户界面)是由若干activity视图页面组成的，我们将每个activity作为一个单元。对于每个activity，测试者在做本地或者远程单元测试的时候，会重复若干次，直到修改完所有的上下文相关单元错误和上下文无关单元错误。

[0034] 具体来说，测试者对app进行单元测试的时候，首先使用开源工具为app编写单元测试脚本，然后使用单元测试脚本在本地移动终端上对app进行单元测试。

[0035] 若app出现错误，则测试者对app进行修改，直到把错误全部修改。

[0036] app单元测试中产生的错误可以分为两类：

[0037] (1)上下文无关错误：在本地移动终端上测试的时候所产生的错误；

[0038] (2)上下文相关错误：在远程云平台移动终端上测试的时候所产生的错误。

[0039] 在本地测试的时候，测试者必须将上下文无关的错误全部改正，否则无法进行后续远程测试。

[0040] 接下来，测试者将单元测试脚本和app部署到云平台上的若干真机上，对app进行远程单元测试。由于单元错误已经被改正，此时若再产生错误，则必定是由于设备差异性产生的，所以是上下文相关错误。这可能是由于云平台上的设备与本地设备在系统版本、屏幕大小、内存大小等方面存在诸多差别而产生的。测试者再次改正上下文相关错误。

[0041] 最后，将本地和远程的单元测试日志保存在测试历史数据库中。

[0042] 接着，在步骤S204中，基于单元测试日志来生成集成测试脚本。

[0043] 图4是示出图2中的步骤S204(即基于单元测试日志来生成集成测试脚本)的一种示例性过程的流程图。

[0044] 如图4所示，在步骤S2042中，首先通过分析单元测试日志来计算各个单元的单元错误率。

[0045] 下面的表1示出了单元测试日志的一个示例。通过分析单元测试日志，可以看到测试者对某个activity做了3次本地测试和2次远程测试。

[0046]

No	测试设备	错误类型	错误数量	总单元错误数
1	本地	上下文无关	8	11.8
2	本地	上下文无关	3	
3	本地	上下文无关	0	
4	远程	上下文相关	2	
5	远程	上下文相关	0	

[0047] 表1

[0048] 在一个示例中，将一个单元的单元错误率定义为该单元的单元测试的错误数与所有单元的单元测试的错误的总数的比率。

[0049] 在本示例中，首先使用下面的公式(1)来计算各个单元的单元测试的错误数。

[0050] $$\text{单元测试的错误数} = \sum_{i=1}^{N_1} \text{上下文无关错误}_i * \gamma^{i-1} + \sum_{i=1}^{N_2} \text{上下文相关关错误}_i * \gamma^{i-1} \quad \ldots (1)$$

[0051] 其中，N_1是本地测试的次数，N_2是远程测试的次数。

[0052] 由于测试者在重复地对同一个单元进行测试的时候，会产生很多相同的错误，所以公式(1)在计算本地单元测试错误数和远程单元测试错误数的时候，使用了折扣率γ进行换算，这种方法避免了同一个单元错误被重复计算。在本示例中，假设γ＝0.6。通过公式(1)可以得出该activity单元的单元错误数为11.8。

[0053] 单元错误率可以通过下面公式(2)来计算。

[0054] $$\text{单元错误率} = \frac{\text{单元错误数}_i}{\sum_{i=1}^{n} \text{单元错误数}_i} \quad \ldots (2)$$

[0055] 其中，n为所有单元的总个数。

[0056] 接下来，在步骤S2044中，基于各个单元的单元错误率来估算各个单元的单元错误概率。

[0057] 在获得单元错误率之后，进一步结合开发人员对app的错误概率的预测来估算各个单元的单元错误概率。在一个示例中，可以用下面的公式(3)来计算单元错误概率：P(单元错误)。

[0058] P(单元错误)＝P(错误预测)*α+P(单元错误率)*(1-α) ...(3)

[0059] 其中，α是权重，范围为0-1。在本示例中，假设α＝0.2。

[0060] 在软件测试领域，有个理论：“在一个软件错误的周围往往隐藏着若干其他错误，错误往往是集中在一起的”。因此，若在某个activity中存在很多单元错误，那么该activity在集成测试的时候，发生集成错误的概率也会较大。此外，app开发者也能预测该

activity的集成错误概率。因此，使用公式(3)来估算集成测试中每个activity的错误发生概率。

[0061] 下面的表2示出了使用公式(3)对集成测试中每个单元的单元错误概率的估算。

[0062]

No	错误预测	单元错误数	单元错误率	单元错误概率
Activity4	0.1	8	0.73	0.604
Activity5	0.25	3	0.27	0.266
Activity6	0.1	0	0	0.02

[0063] 表2

[0064] 从表2可以看到，Activity4的单元错误概率最高，为0.604。

[0065] 接下来，在步骤S2046中，基于各个单元的单元错误概率生成集成测试路径。

[0066] 在一个示例中，通过将应用程序的源代码转换为所有单元的图结构，优先选择单元错误概率最大的单元遍历图结构来生成集成测试路径。

[0067] 图5示出了集成测试路径生成的一个示例过程。首先，通过分析并且解析android app的配置文件来获取app的图结构。图结构有效地描述了app包含的所有activity的层次结构，因此我们可以利用图结构来生成集成测试路径。例如，图5中，当测试路径到达结点activity2的时候，此时下一个结点选择哪个呢?由于在上面的表2中示出，activity4的预测错误概率最高，为0.604，因此测试路径会选择activity4。使用这种方法，根据本发明的app测试方法基于错误概率贪婪，生成最终的测试路径。即从app的activity根结点开始遍历图结构，遇到分支的时候，每次都选择错误概率最高的那个结点。从而生成了集成测试路径。

[0068] 接着，在步骤S2048中，利用所述集成测试路径来生成集成测试脚本。我们将使用测试数据自动适配测试路径的方法来生成最终的测试脚本。

[0069] 在一个示例中，基于模型－视图－控制器(MVC)模式，通过循环探测生成满足集成测试路径上的单元跳转条件的测试数据，根据测试数据和集成测试路径来生成集成测试脚本。

[0070] MVC模式包括：模型层(Model)：测试数据自动适配测试路径，视图层(View)：集成测试的截图，以及控制层(Control)：集成测试脚本。

[0071] 具体地，首先，根据测试路径，分析activity跳转所绑定的事件自动地生成测试脚本模板，然后将测试路径与测试数据分离，使用不同的测试数据去填充测试模板。若要引发Activity之间的跳转，则必须用满足要求的测试数据去填充测试模板。图6示出了基于MVC的集成测试运行的一个示例。如图6所示，若activity2想要跳转到activity4，跳转参数格式在图6的左半部分，通过搜索测试输入数据库寻找满足条件的测试数据，从而填充测试模板，形成不同的测试脚本。最终使用该测试脚本对app进行集成测试。基于MVC模式，可以方便测试者直观地观察测试结果，并且可以使用不同的测试数据快速生成不同的测试脚本。

[0072] 图7示出了基于MVC进行集成测试的示意图。

[0073] 集成云测试方法与单元云测试方法几乎相同，首先我们在本地设备上对app进行集成测试，接着选择远程设备对app进行集成测试，例如，我们选择Device1，Device2，Device3和Device4，然后将app部署到远程设备上，对其测试。最终，我们通过比较本地测试的日志和远程测试的日志来判断测试结果，本地测试日志被作为正确的结果进行参考，若远程测试日志与本地测试日志相同，则远程测试结果正确，否则远程测试结果失败。如图7所示，示出了集成测试的结果和每个远程测试设备的上下文参数配置。在图7中，上下文＝(内存，OS版本，屏幕大小，gps)。

[0074] 在集成测试结束后，可以发现app在Device1，Device2和Device3上的测试结果是正确的，在Device4上的测试结果错误。因此，Device1，Device2和Device3是适配于被测试的app。根据本发明的一个实施例的对移动终端上的应用程序进行测试的方法能够进一步利用集成测试结果，进行候选移动终端与应用程序的适配性测试，从而推荐其它的适配手机给测试者。由于在远程部署所有类型的手机真机无法实现，因此使用该方法可以有效地找出所有适配于app的手机类型。

[0075] 可以在根据本发明的一个实施例的对移动终端上的应用程序进行测试的装置中提供存储所有手机配置参数的数据库，将数据库中的手机作为候选适配手机。首先，从数据中获取一个候选手机的配置参数，然后计算候选手机与测试手机之间的设备相似度，从而选出和候选手机具有相似配置的测试手机。最终，通过计算相似测试手机的测试正确率来判断该候选手机是否适配于app。简单来说，我们认为，当测试手机的测试结果为Right(正确)时，与其具有较高相似度的手机也有较高的概率与该app适配。图8示出了对适配手机类型进行推荐的一个示例过程。

[0076] 首先，将每个测试手机的上下文参数归一化。例如：候选设备的初始上下文参数$Device_{candidate_original}$＝(4,4.2,1136,700,1)。归一化的上下文参数$Device_{candidate}$＝(0.5,0.96,0.62,1)。

[0077] 然后，我们使用公式(4)计算候选手机与每个测试手机的相似度。

[0078] 相似度$(Device_{candidate}, Device_i) = \frac{Device_{candidate} * Device_i}{|Device_{candidate}||Device_i|}$ (4)

[0079] 其中，i＝1,2…

[0080] 在这里，假定$Memory_{max}$＝8，$version_{max}$＝4.5。

[0081] 下面的表3示出了根据公式(4)计算出的设备相似度的计算结果。

[0082]

设备	上下文归一化	测试结果	相似度
Device1	(0.25,0.91,0.67,1)	Right	0.9872
Device2	(0.375,0.93,0.75,1)	Right	0.9717
Device3	(0.5,0.93,0.56,1)	Right	0.9919
Device4	(0.5,0.96,0.56,1)	Wrong	0.9993

[0083] 表3

[0084] 测试结果为Right表示正确，Wrong表示错误。

[0085] 如果设定阈值为0.98，可以看到，表3中的Device1，Device3和Device4符合要求，被认为是与候选手机相似的设备。

[0086] 接下来使用app在相似测试手机上的集成测试结果，来预测候选手机是否适配于app。

[0087] 在一个示例中，可以使用下面的公式5，基于条件概率计算候选手机的适配概率。若适配概率大于阈值，则认为该候选手机适配于app，否则认为其不适配于app。

[0088] $$P(right \mid device_{candidate}) = \frac{P(right * Device = device_1, device_3, device_4)}{P(context = context_{candidate})} = \frac{P(right * Device = device_1, device_3, device_4)}{P(Device = device_1, device_3, device_4)}$$

[0089] 根据表1中的数据，可以得到$P(right \mid device_{candidateDevice1}) = \frac{2/3}{3/4} = 0.89$。

[0090] 在本例中，使用这种计算方法，获得了candidateDevice1的适配概率(集成测试正确概率)。假设有3个候选手机，如表4所示，使用同样的方法分别计算candidateDevice2和candidateDevice3的适配概率。若阈值为0.85。则最终candidateDevice1和candidateDevice2满足要求，作为最终的适配手机，推荐给测试者。

[0091]

适配设备候选	正确概率	阈值	推荐设备列表
candidateDevice1	0.89	0.85	candidateDevice1, candidateDevice2
candidateDevice2	0.86		

[0092]

candidateDevice3	0.81		

[0093] 表4

[0094] 接下来，将参照图9-12描述根据本发明的一个实施例的移动终端上的应用程序的测试装置。

[0095] 图9是示出根据本发明的一个实施例的移动终端上的应用程序的测试装置900的示例性配置的框图。

[0096] 如图9所示，移动终端上的应用程序的测试装置900包括：单元测试单元902、集成测试脚本生成单元904、集成测试单元906、以及测试结果提供单元908。

[0097] 其中，单元测试单元902被配置为使用单元测试脚本分别在本地移动终端上和布置在云上的远程移动终端上对应用程序执行单元测试，以得到所述应用程序的各个单元的单元测试日志。

[0098] 集成测试脚本生成单元904被配置为基于所述单元测试日志来生成集成测试脚本。

[0099] 集成测试单元906被配置为使用所述集成测试脚本分别在所述本地移动终端和所述远程移动终端上对所述应用程序执行集成测试。

[0100] 测试结果提供单元908被配置为通过比较本地移动终端的集成测试日志和远程移动终端的集成测试日志来提供集成测试结果。

[0101] 图10示出根据本发明的另一个实施例的移动终端上的应用程序的测试装置900'的另一种示例性配置的框图。

[0102] 测试装置900'除了包括图9所示的单元测试单元902、集成测试脚本生成单元904、集成测试单元906、以及测试结果提供单元908之外，还包括移动终端适配性测试单元910。

[0103] 已参照图9给出了关于单元测试单元902、集成测试脚本生成单元904、集成测试单元906、以及测试结果提供单元908的详细描述，在此不做赘述。将具体说明移动终端适配性测试单元910。

[0104] 移动终端适配性测试单元910被配置为利用所述集成测试结果，进行候选移动终端与所述应用程序的适配性测试。从而可以向测试者推荐其它的适配手机。

[0105] 图11是示出图9中的集成测试脚本生成单元904的一种示例性配置的框图。

[0106] 如图11所示，集成测试脚本生成单元904包括单元错误率计算子单元9042、单元错误概率计算子单元9044、集成测试路径生成子单元9046和集成测试脚本生成子单元9048。

[0107] 单元错误率计算子单元9042被配置为通过分析所述单元测试日志来计算各个单元的单元错误率。

[0108] 单元错误概率计算子单元9044被配置为至少基于各个单元的所述单元错误率来估算各个单元的单元错误概率。

[0109] 集成测试路径生成子单元9046被配置为基于各个单元的所述单元错误概率生成集成测试路径。

[0110] 集成测试脚本生成子单元9048被配置为利用所述集成测试路径来生成所述集成测试脚本。

[0111] 其中所述单元错误率是该单元的单元测试的错误数与全部单元的单元测试的错误的总数的比率。

[0112] 其中，所述单元测试的错误数通过如下方式计算：将在本地移动终端上对所述单元执行的每次单元测试中的错误的数目分别乘以不同的系数再求和得到第一错误数，将在远程移动终端上对所述单元执行的每次单元测试的错误的数目分别乘以不同系数再求和得到第二错误数，所述单元测试的错误数为第一错误数与第二错误数之和。

[0113] 所述单元错误概率计算子单元9044进一步被配置为：基于各个单元的单元错误率和预测错误概率来加权计算各个单元的单元错误概率。

[0114] 集成测试路径生成子单元9046进一步被配置为：将所述应用程序的源代码转换为所有单元的图结构，优先选择单元错误概率最大的单元遍历所述图结构来生成集成测试路径。

[0115] 集成测试脚本生成子单元9048进一步被配置为：基于模型－视图－控制器模式，通过循环探测生成满足所述集成测试路径上的单元跳转条件的测试数据，根据所述测试数据和所述集成测试路径来生成集成测试脚本。

[0116] 图12是示出图10中的移动终端适配性测试单元910的一种示例性配置的框图。

[0117] 移动终端适配性测试单元910包括：相似度计算子单元9102、适配概率计算子单元9104、和适配终端选择子单元9106。

[0118] 其中，相似度计算子单元9102被配置为计算所述候选移动终端与执行所述集成测试的测试移动终端之间的相似度；

[0119] 适配概率计算子单元9104被配置为选择与所述候选移动终端的相似度大于预定阈值的测试移动终端的集成测试结果来计算所述候选移动终端与所述应用程序的适配概率;以及

[0120] 适配终端选择子单元9106被配置为选择与所述应用程序的所述适配概率大于预定阈值的候选移动终端作为适配终端。

[0121] 关于移动终端上的应用程序的测试装置900的各个部分的操作和功能的细节可以参照结合图1-8描述的本发明的移动终端上的应用程序的测试方法的实施例,这里不再详细描述。

[0122] 在此需要说明的是,图9-12所示的移动终端上的应用程序的测试装置及其组成单元的结构仅仅是示例性的,本领域技术人员可以根据需要对图9-12所示的结构框图进行修改。

[0123] 本发明提出一种移动终端上的应用程序的测试方法和装置,其具有以下技术优势:

[0124] 提供了一种手机app云测试的框架。在云平台上分布式地部署相当数量的手机,测试者基于云平台,选择不同的手机,上传app和测试脚本,对app进行设备多样性测试。

[0125] 基于单元测试历史数据和app的UI配置文件自动化生成集成测试脚本。测试框架在遍历app的图结构时,基于单元测试历史数据和测试者的概率估算,对每个activity节点进行错误率预测,从而选择错误率最高的节点,生成测试路径。并且,框架通过循环探测activity参数的方法,生成满足测试路径的测试数据,最终生成集成测试脚本。

[0126] 基于MVC模式的集成测试。测试框架将测试数据,测试脚本的执行,查看测试结果3个模块分离,基于MVC模式对app进行集成测试,从而方便测试者生成不同组合的测试脚本,并且便于对app进行测试结果的观察。

[0127] 适配于app的手机类型推荐。根据集成测试的结果,测试框架通过分析与候选设备相似的测试设备的测试结果,来对候选设备是否适配于app进行评价,最终推荐出所有适配于该app的手机类型。通过这种方法,可以对云平台上没有的真机进行适配计算。

[0128] 测试者可以将app部署在远程不同种类的模拟器和手机真机上,然后进行设备多样性的差异测试。通过这种方法,测试者不需要购买任何真机,因此也节省了很多成本。测试框架利用单元测试错误率和开发者的错误预测概率来自动化地生成测试路径。此外,框架将测试模板和测试数据分离,自动化地探测满足测试路径的测试输入数据,从而生成测试脚本。最后,框架基于MVC模式自动化地测试app。框架通过分析与候选设备相似的测试设备的集成测试结果,来判断候选设备是否适配于app。最终,推荐出所有的适配手机类型给测试者。

[0129] 以上结合具体实施例描述了本发明的基本原理,但是,需要指出的是,对本领域的普通技术人员而言,能够理解本发明的方法和装置的全部或者任何步骤或者部件,可以在任何计算装置(包括处理器、存储介质等)或者计算装置的网络中,以硬件、固件、软件或者它们的组合加以实现,这是本领域普通技术人员在阅读了本发明的说明的情况下运用他们的基本编程技能就能实现的。

[0130] 因此,本发明的目的还可以通过在任何计算装置上运行一个程序或者一组程序来实现。所述计算装置可以是公知的通用装置。因此,本发明的目的也可以仅仅通过提供包含

实现所述方法或者装置的程序代码的程序产品来实现。也就是说，这样的程序产品也构成本发明，并且存储有这样的程序产品的存储介质也构成本发明。显然，所述存储介质可以是任何公知的存储介质或者将来所开发出来的任何存储介质。

[0131] 在通过软件和/或固件实现本发明的实施例的情况下，从存储介质或网络向具有专用硬件结构的计算机，例如图13所示的通用计算机1300安装构成该软件的程序，该计算机在安装有各种程序时，能够执行各种功能等等。

[0132] 在图13中，中央处理单元(CPU)1301根据只读存储器(ROM)1302中存储的程序或从存储部分1308加载到随机存取存储器(RAM)1303的程序执行各种处理。在RAM 1303中，也根据需要存储当CPU 1301执行各种处理等等时所需的数据。CPU 1301、ROM 1302和RAM 1303经由总线1304彼此链路。输入/输出接口1305也链路到总线1304。

[0133] 下述部件链路到输入/输出接口1305：输入部分1306(包括键盘、鼠标等等)、输出部分1307(包括显示器，比如阴极射线管(CRT)、液晶显示器(LCD)等，和扬声器等)、存储部分1308(包括硬盘等)、通信部分1309(包括网络接口卡比如LAN卡、调制解调器等)。通信部分1309经由网络比如因特网执行通信处理。根据需要，驱动器1310也可链路到输入/输出接口1305。可拆卸介质1311比如磁盘、光盘、磁光盘、半导体存储器等等根据需要被安装在驱动器1310上，使得从中读出的计算机程序根据需要被安装到存储部分1308中。

[0134] 在通过软件实现上述系列处理的情况下，从网络比如因特网或存储介质比如可拆卸介质1311安装构成软件的程序。

[0135] 本领域的技术人员应当理解，这种存储介质不局限于图13所示的其中存储有程序、与设备相分离地分发以向用户提供程序的可拆卸介质1311。可拆卸介质1311的例子包含磁盘(包含软盘(注册商标))、光盘(包含光盘只读存储器(CD-ROM)和数字通用盘(DVD))、磁光盘(包含迷你盘(MD)(注册商标))和半导体存储器。或者，存储介质可以是ROM 1302、存储部分1308中包含的硬盘等等，其中存有程序，并且与包含它们的设备一起被分发给用户。

[0136] 本发明还提出一种存储有机器可读取的指令代码的程序产品。指令代码由机器读取并执行时，可执行上述根据本发明实施例的方法。

[0137] 相应地，用于承载上述存储有机器可读取的指令代码的程序产品的存储介质也包括在本发明的公开中。存储介质包括但不限于软盘、光盘、磁光盘、存储卡、存储棒等。

[0138] 本领域的普通技术人员应理解，在此所例举的是示例性的，本发明并不局限于此。

[0139] 在本说明书中，“第一”、“第二”以及“第N个”等表述是为了将所描述的特征在文字上区分开，以清楚地描述本发明。因此，不应将其视为具有任何限定性的含义。

[0140] 作为一个示例，上述方法的各个步骤以及上述设备的各个组成模块和/或单元可以实施为软件、固件、硬件或其组合，并作为相应设备中的一部分。上述装置中各个组成模块、单元通过软件、固件、硬件或其组合的方式进行配置时可使用的具体手段或方式为本领域技术人员所熟知，在此不再赘述。

[0141] 作为一个示例，在通过软件或固件实现的情况下，可以从存储介质或网络向具有专用硬件结构的计算机(例如图13所示的通用计算机1300)安装构成该软件的程序，该计算机在安装有各种程序时，能够执行各种功能等。

[0142] 在上面对本发明具体实施例的描述中，针对一种实施方式描述和/或示出的特征可以以相同或类似的方式在一个或更多个其他实施方式中使用，与其他实施方式中的特征

相组合，或替代其他实施方式中的特征。

[0143] 应该强调，术语“包括/包含”在本文使用时指特征、要素、步骤或组件的存在，但并不排除一个或更多个其他特征、要素、步骤或组件的存在或附加。

[0144] 此外，本发明的方法不限于按照说明书中描述的时间顺序来执行，也可以按照其他的时间顺序地、并行地或独立地执行。因此，本说明书中描述的方法的执行顺序不对本发明的技术范围构成限制。

[0145] 本发明及其优点，但是应当理解在不超出由所附的权利要求所限定的本发明的精神和范围的情况下可以进行各种改变、替代和变换。而且，本发明的范围不仅限于说明书所描述的过程、设备、手段、方法和步骤的具体实施例。本领域内的普通技术人员从本发明的公开内容将容易理解，根据本发明可以使用执行与在此的相应实施例基本相同的功能或者获得与其基本相同的结果的、现有和将来要被开发的过程、设备、手段、方法或者步骤。因此，所附的权利要求旨在在它们的范围内包括这样的过程、设备、手段、方法或者步骤。

[0146] 基于以上的说明，可知公开至少公开了以下技术方案：

[0147] 附记1、一种移动终端上的应用程序的测试方法，包括：

[0148] 使用单元测试脚本分别在本地移动终端上和布置在云上的远程移动终端上对应用程序执行单元测试，以得到所述应用程序的各个单元的单元测试日志；

[0149] 基于所述单元测试日志来生成集成测试脚本；

[0150] 使用所述集成测试脚本分别在所述本地移动终端和所述远程移动终端上对所述应用程序执行集成测试；以及

[0151] 通过比较本地移动终端的集成测试日志和远程移动终端的集成测试日志来提供集成测试结果。

[0152] 附记2、根据附记1所述的方法，还包括：利用所述集成测试结果，进行候选移动终端与所述应用程序的适配性测试。

[0153] 附记3、根据附记1所述的方法，其中，基于所述单元测试日志来生成集成测试脚本包括：

[0154] 通过分析所述单元测试日志来计算各个单元的单元错误率；

[0155] 至少基于各个单元的所述单元错误率来估算各个单元的单元错误概率；

[0156] 基于各个单元的所述单元错误概率生成集成测试路径；以及

[0157] 利用所述集成测试路径来生成所述集成测试脚本。

[0158] 附记4、根据附记3所述的方法，其中所述单元错误率是该单元的单元测试的错误数与全部单元的单元测试的错误的总数的比率；

[0159] 其中，所述单元测试的错误数通过如下方式计算：将在本地移动终端上对所述单元执行的每次单元测试中的错误的数目分别乘以不同的系数再求和得到第一错误数，将在远程移动终端上对所述单元执行的每次单元测试的错误的数目分别乘以不同系数再求和得到第二错误数，所述单元测试的错误数为第一错误数与第二错误数之和。

[0160] 附记5、根据附记3所述的方法，其中，至少基于各个单元的单元错误率来计算各个单元的单元错误概率进一步包括：

[0161] 基于各个单元的单元错误率和错误预测概率来加权计算各个单元的单元错误概率。

[0162] 附记6、根据附记3所述的方法，其中，基于各个单元的单元错误概率生成集成测试路径包括：

[0163] 将所述应用程序的源代码转换为所有单元的图结构，优先选择单元错误概率最大的单元遍历所述图结构来生成集成测试路径。

[0164] 附记7、根据附记3所述的方法，其中，利用所述集成测试路径来生成集成测试脚本包括：

[0165] 基于模型－视图－控制器模式，通过循环探测生成满足所述集成测试路径上的单元跳转条件的测试数据，根据所述测试数据和所述集成测试路径来生成集成测试脚本。

[0166] 附记8、根据附记1所述的方法，其中，将所述应用程序的一个视图页面作为一个单元。

[0167] 附记9、根据附记2所述的方法，其中，利用所述集成测试结果，进行候选移动终端与所述应用程序的适配性测试包括：

[0168] 计算所述候选移动终端与执行所述集成测试的测试移动终端之间的相似度；

[0169] 选择与所述候选移动终端的相似度大于第一预定阈值的测试移动终端的集成测试结果来计算所述候选移动终端与所述应用程序的适配概率；以及

[0170] 选择与所述应用程序的所述适配概率大于第二预定阈值的候选移动终端作为适配终端。

[0171] 附记10、一种移动终端上的应用程序的测试装置，包括：

[0172] 单元测试单元，被配置为使用单元测试脚本分别在本地移动终端上和布置在云上的远程移动终端上对应用程序执行单元测试，以得到所述应用程序的各个单元的单元测试日志；

[0173] 集成测试脚本生成单元，被配置为基于所述单元测试日志来生成集成测试脚本；

[0174] 集成测试单元，被配置为使用所述集成测试脚本分别在所述本地移动终端和所述远程移动终端上对所述应用程序执行集成测试；以及

[0175] 测试结果提供单元，被配置为通过比较本地移动终端的集成测试日志和远程移动终端的集成测试日志来提供集成测试结果。

[0176] 附记11、根据附记10所述的装置，还包括：移动终端适配性测试单元，被配置为利用所述集成测试结果，进行候选移动终端与所述应用程序的适配性测试。

[0177] 附记12、根据附记10所述的装置，其中，所述集成测试脚本生成单元包括：

[0178] 单元错误率计算子单元，被配置为通过分析所述单元测试日志来计算各个单元的单元错误率；

[0179] 单元错误概率计算子单元，被配置为至少基于各个单元的所述单元错误率来估算各个单元的单元错误概率；

[0180] 集成测试路径生成子单元，被配置为基于各个单元的所述单元错误概率生成集成测试路径；以及

[0181] 集成测试脚本生成子单元，被配置为利用所述集成测试路径来生成所述集成测试脚本。

[0182] 附记13、根据附记12所述的装置，其中所述单元错误率是该单元的单元测试的错误数与全部单元的单元测试的错误的总数的比率；

[0183] 其中，所述单元测试的错误数通过如下方式计算：将在本地移动终端上对所述单元执行的每次单元测试中的错误的数目分别乘以不同的系数再求和得到第一错误数，将在远程移动终端上对所述单元执行的每次单元测试的错误的数目分别乘以不同系数再求和得到第二错误数，所述单元测试的错误数为第一错误数与第二错误数之和。

[0184] 附记14、根据附记12所述的装置，其中，所述单元错误概率计算子单元进一步被配置为：

[0185] 基于各个单元的单元错误率和错误预测概率来加权计算各个单元的单元错误概率。

[0186] 附记15、根据附记12所述的装置，其中，所述集成测试路径生成单元进一步被配置为：

[0187] 将所述应用程序的源代码转换为所有单元的图结构，优先选择单元错误概率最大的单元遍历所述图结构来生成集成测试路径。

[0188] 附记16、根据附记12所述的装置，其中，所述集成测试脚本生成单元进一步被配置为：

[0189] 基于模型－视图－控制器模式，通过循环探测生成满足所述集成测试路径上的单元跳转条件的测试数据，根据所述测试数据和所述集成测试路径来生成集成测试脚本。

[0190] 附记17、根据附记10所述的装置，其中，将所述应用程序的一个视图页面作为一个单元。

[0191] 附记18、根据附记11所述的装置，其中，所述移动终端适配性测试单元包括：

[0192] 相似度计算子单元，被配置为计算所述候选移动终端与执行所述集成测试的测试移动终端之间的相似度；

[0193] 适配概率计算子单元，被配置为选择与所述候选移动终端的相似度大于第一预定阈值的测试移动终端的集成测试结果来计算所述候选移动终端与所述应用程序的适配概率；以及

[0194] 适配终端选择子单元，被配置为选择与所述应用程序的所述适配概率大于第二预定阈值的候选移动终端作为适配终端。

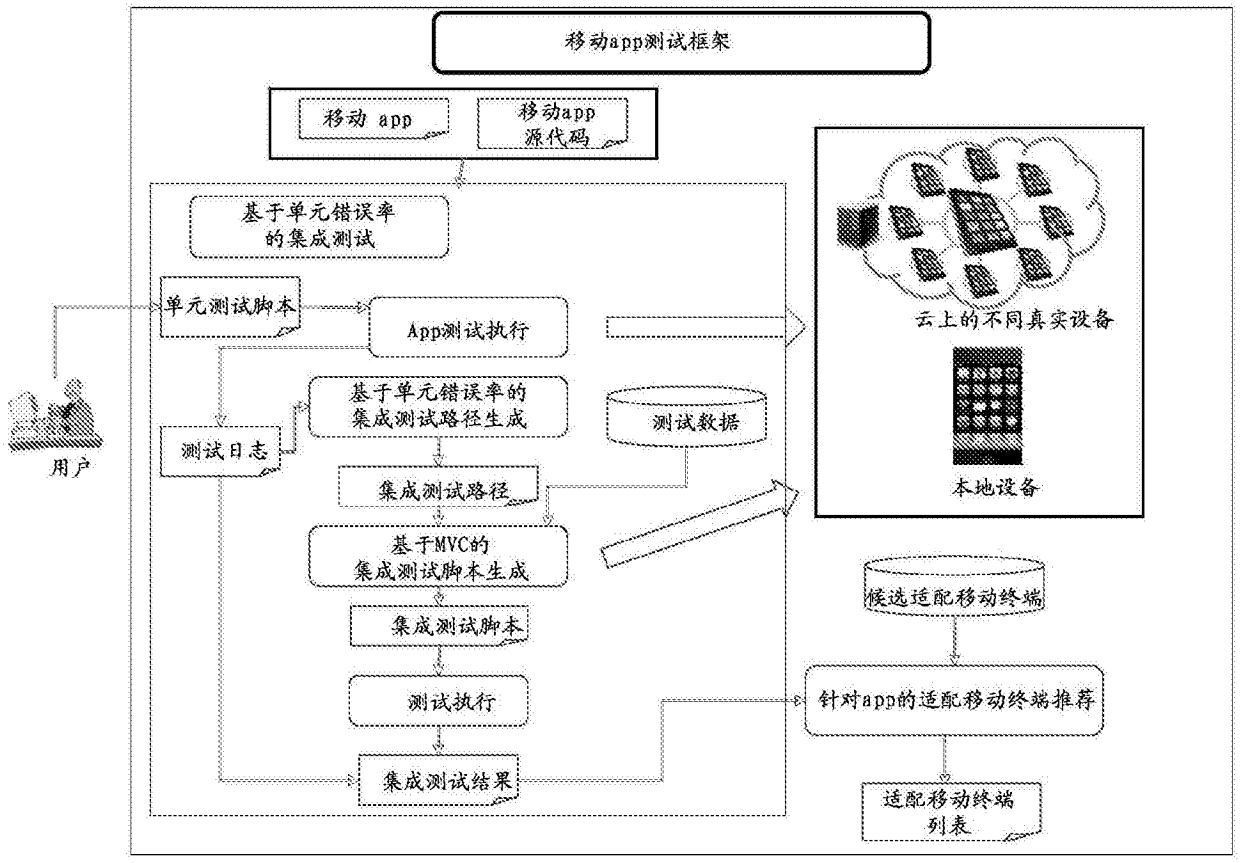

移动app测试框架
移动 app
移动app
源代码
基于单元错误率
的集成测试
单元测试脚本
App测试执行
基于单元错误率的
集成测试路径生成
测试数据
测试日志
集成测试路径
基于MVC的
集成测试脚本生成
集成测试脚本
测试执行
集成测试结果
用户
云上的不同真实设备
本地设备
候选适配移动终端
针对app的适配移动终端推荐
适配移动终端
列表

图1

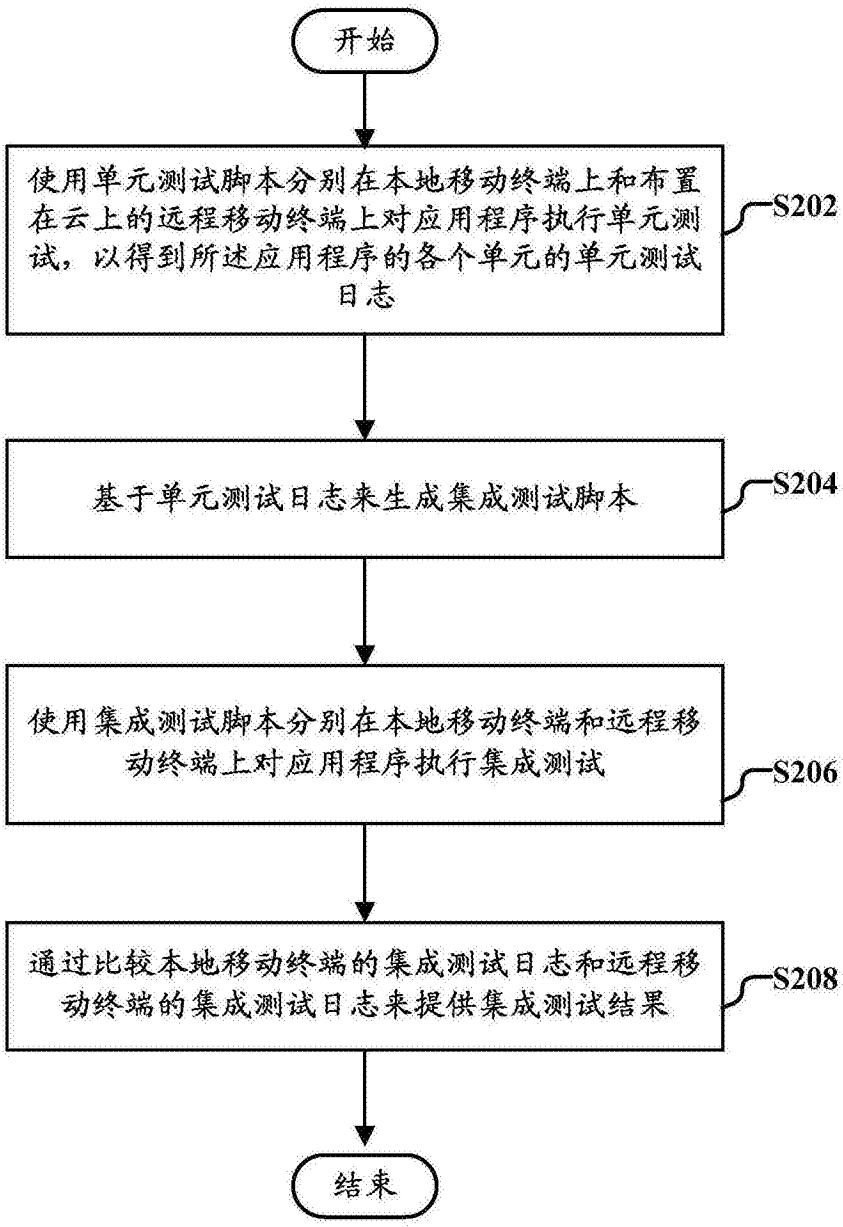

开始
使用单元测试脚本分别在本地移动终端上和布置在云上的远程移动终端上对应用程序执行单元测试，以得到所述应用程序的各个单元的单元测试日志
S202
基于单元测试日志来生成集成测试脚本
S204
使用集成测试脚本分别在本地移动终端和远程移动终端上对应用程序执行集成测试
S206
通过比较本地移动终端的集成测试日志和远程移动终端的集成测试日志来提供集成测试结果
S208
结束

图2

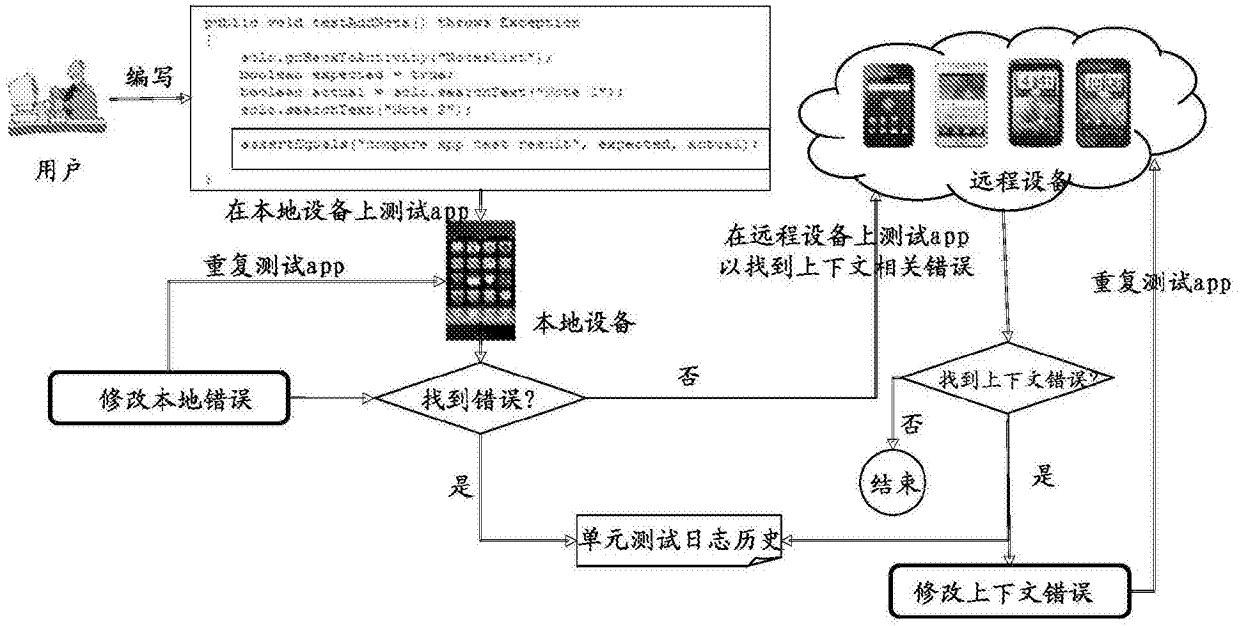

编写
用户
在本地设备上测试app
远程设备
在远程设备上测试app
以找到上下文相关错误
重复测试app
重复测试app
本地设备
修改本地错误
找到错误?
否
找到上下文错误?
否
结束
是
是
单元测试日志历史史
修改上下文错误

图3

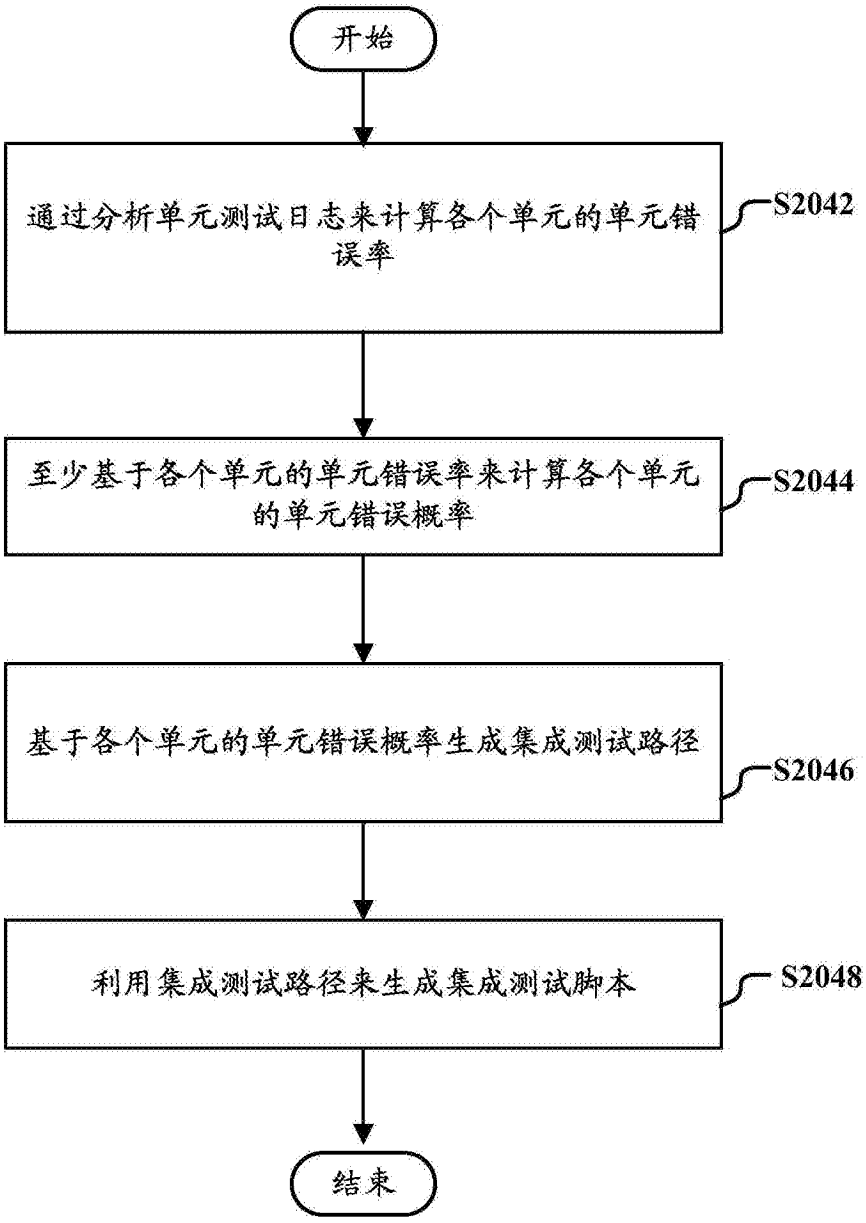

开始
通过分析单元测试日志来计算各个单元的单元错误率
S2042
至少基于各个单元的单元错误率来计算各个单元的单元错误概率
S2044
基于各个单元的单元错误概率生成集成测试路径
S2046
利用集成测试路径来生成集成测试脚本
S2048
结束

图4

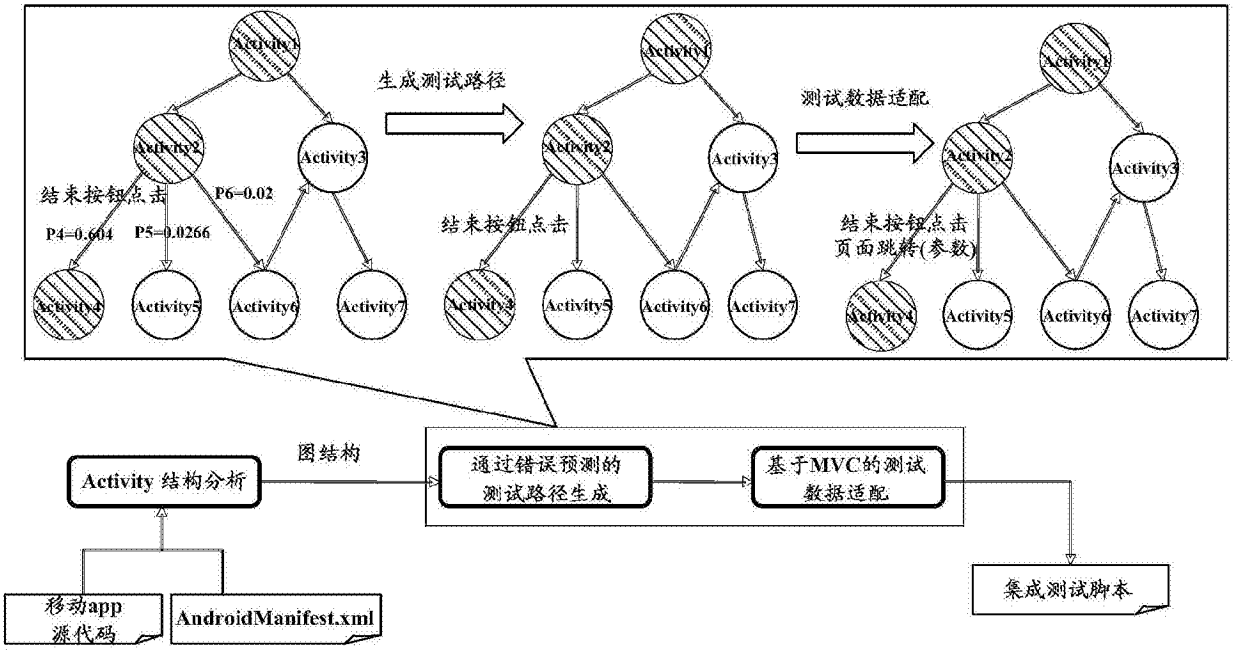

Activity1
Activity2
Activity3
Activity4
Activity5
Activity6
Activity7
生成测试路径
测试数据适配
结束按钮点击
P6=0.02
P4=0.604
P5=0.0266
结束按钮点击
结束按钮点击
页面跳转(参数)
Activity 结构分析
图结构
通过错误预测的
测试路径生成
基于MVC的测试
数据适配
移动app
源代码
AndroidManifest.xml
集成测试脚本

图5

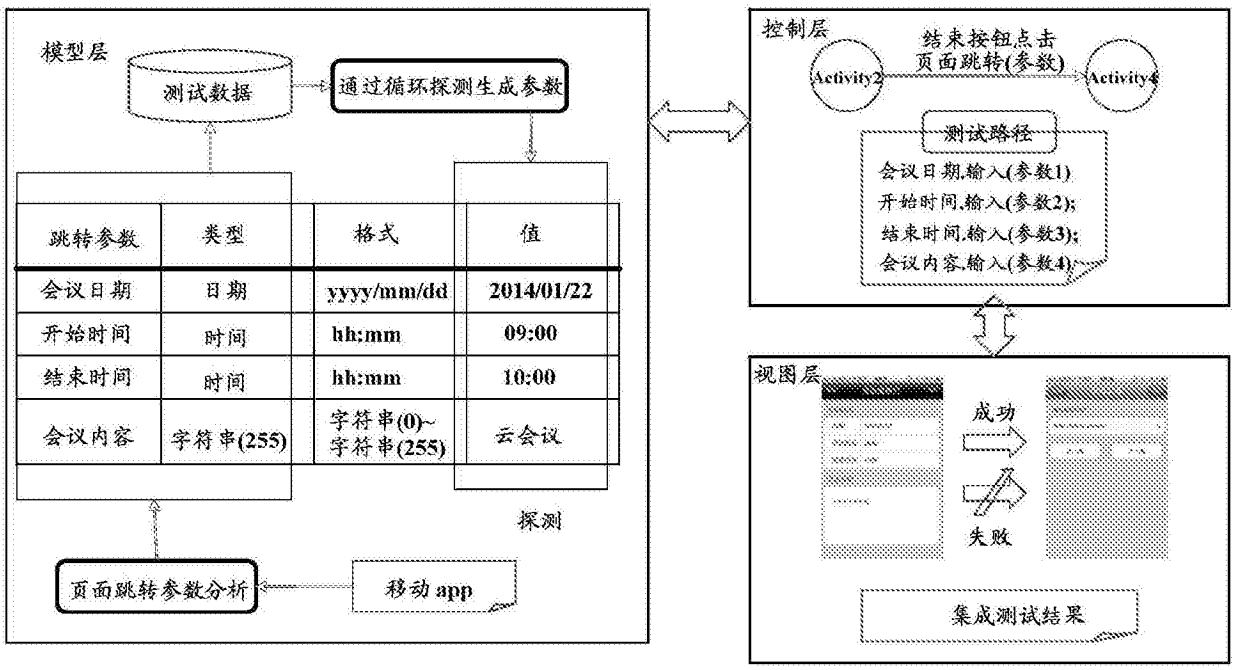
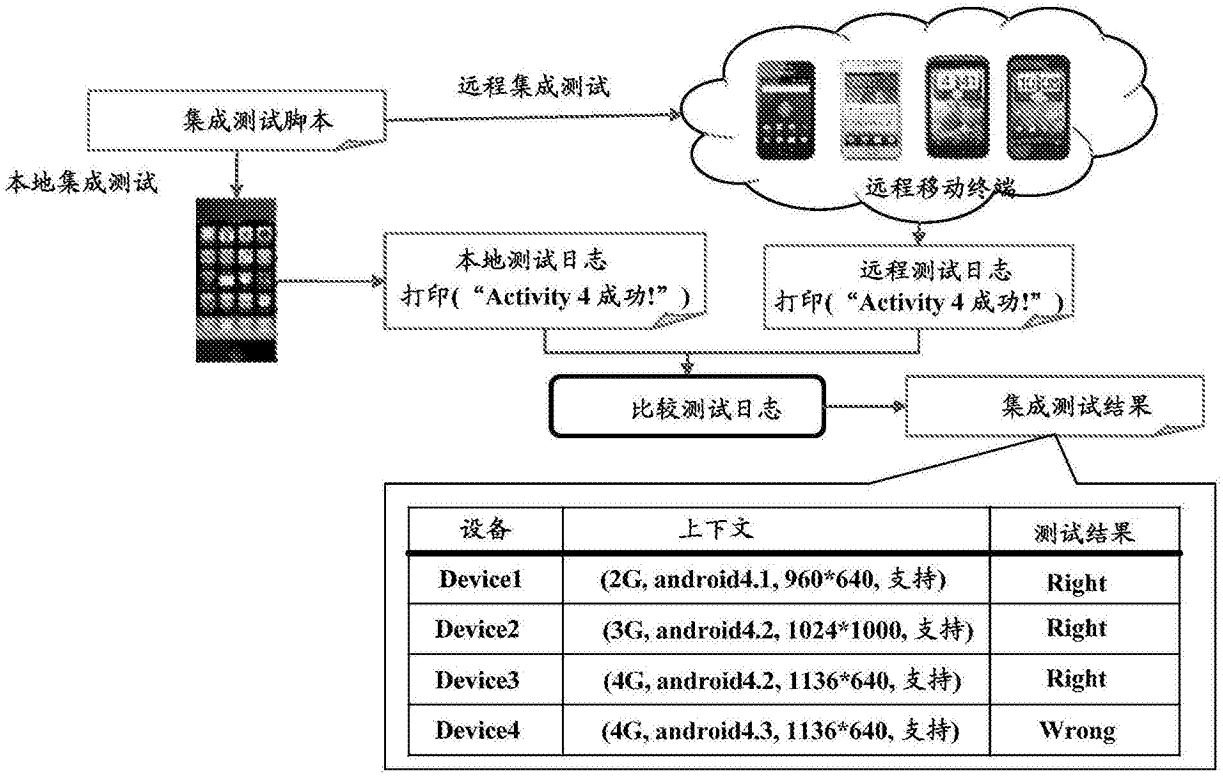

模型层
测试数据
通过循环探测生成参数
跳转参数
类型
格式
值
会议日期
日期
yyyy/mm/dd
2014/01/22
开始时间
时间
hh:mm
09:00
结束时间
时间
hh:mm
10:00
会议内容
字符串(255)
字符串(0)~
字符串(255)
云会议
探测
页面跳转参数分析
移动 app
控制层
Activity2
结束按钮点击
页面跳转(参数)
Activity4
测试路径
会议日期,输入(参数1)
开始时间,输入(参数2);
结束时间,输入(参数3);
会议内容,输入(参数4);
视图层
成功
失败
集成测试结果

图6



设备	上下文	测试结果
Device1	(2G, android4.1, 960*640, 支持)	Right
Device2	(3G, android4.2, 1024*1000, 支持)	Right
Device3	(4G, android4.2, 1136*640, 支持)	Right
Device4	(4G, android4.3, 1136*640, 支持)	Wrong

图7

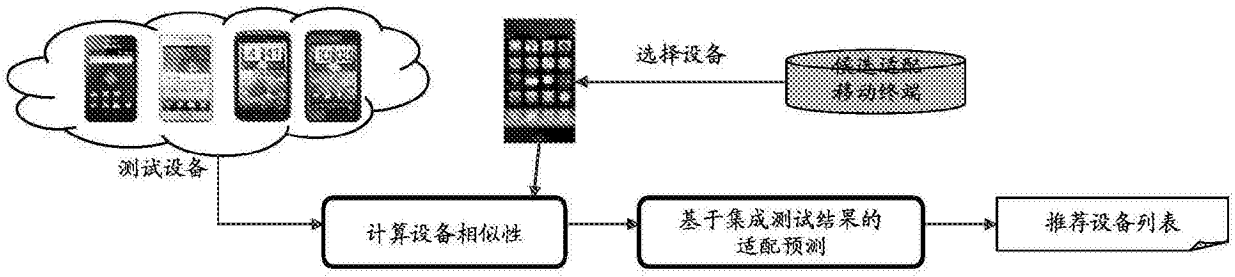


图8

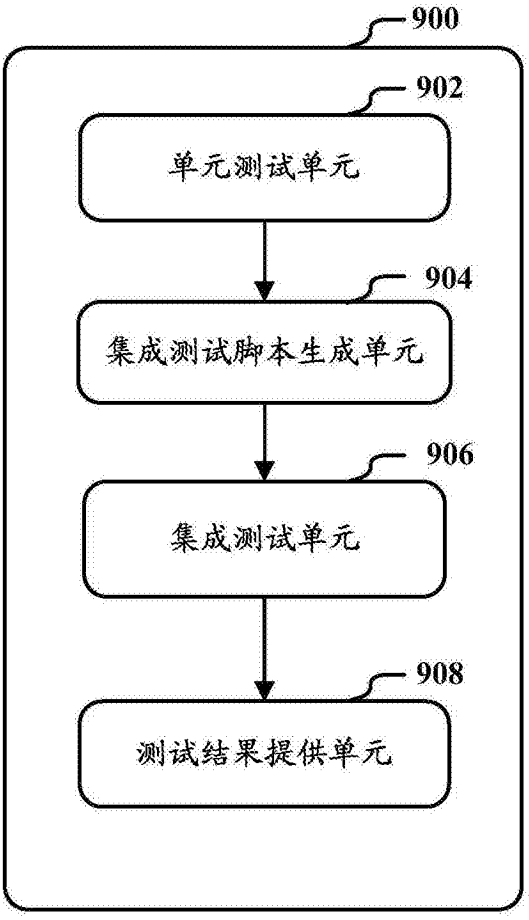


图9

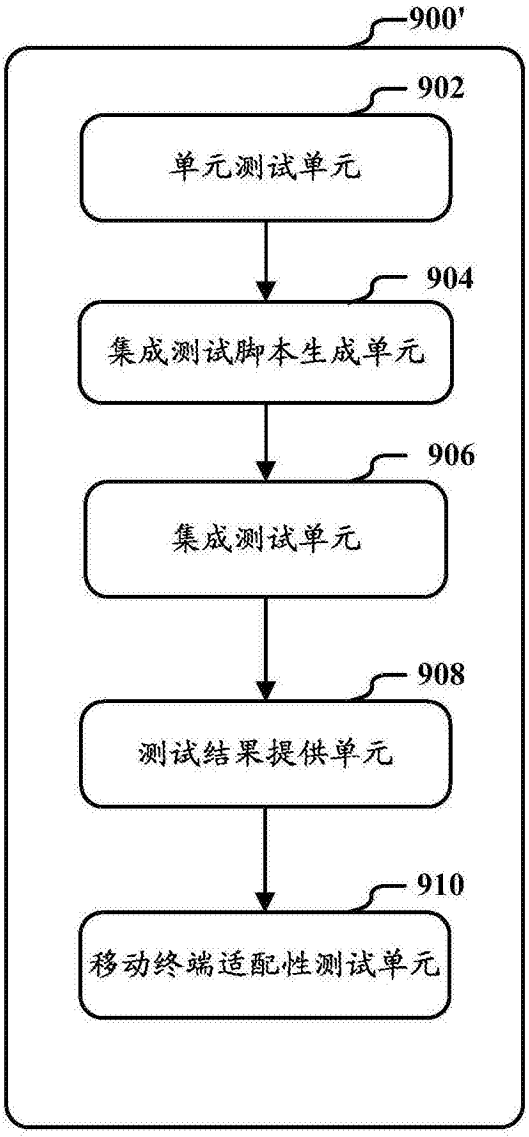

900'
902
单元测试单元
904
集成测试脚本生成单元
906
集成测试单元
908
测试结果提供单元
910
移动终端适配性测试单元

图10

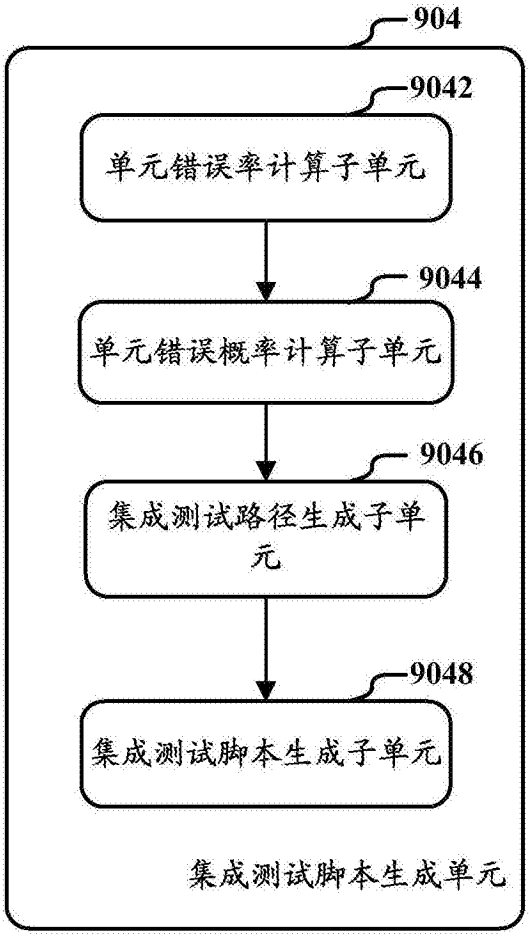

904
9042
单元错误率计算子单元
9044
单元错误概率计算子单元
9046
集成测试路径生成子单元
9048
集成测试脚本生成子单元
集成测试脚本生成单元

图11

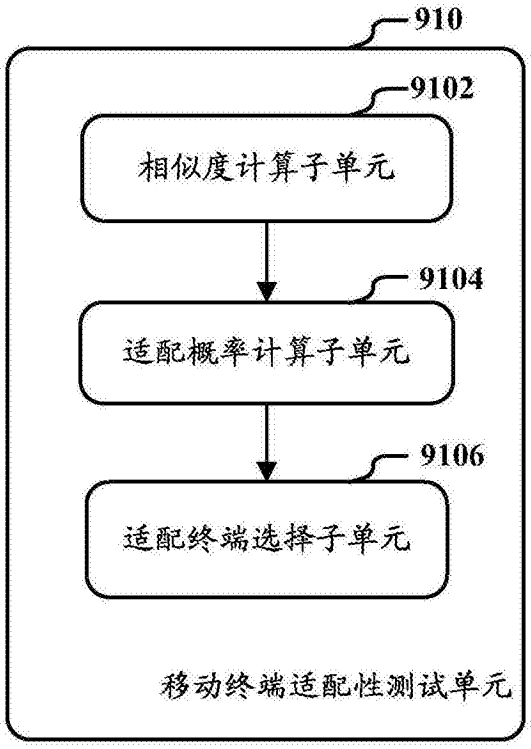

910
9102
相似度计算子单元
9104
适配概率计算子单元
9106
适配终端选择子单元
移动终端适配性测试单元

图12

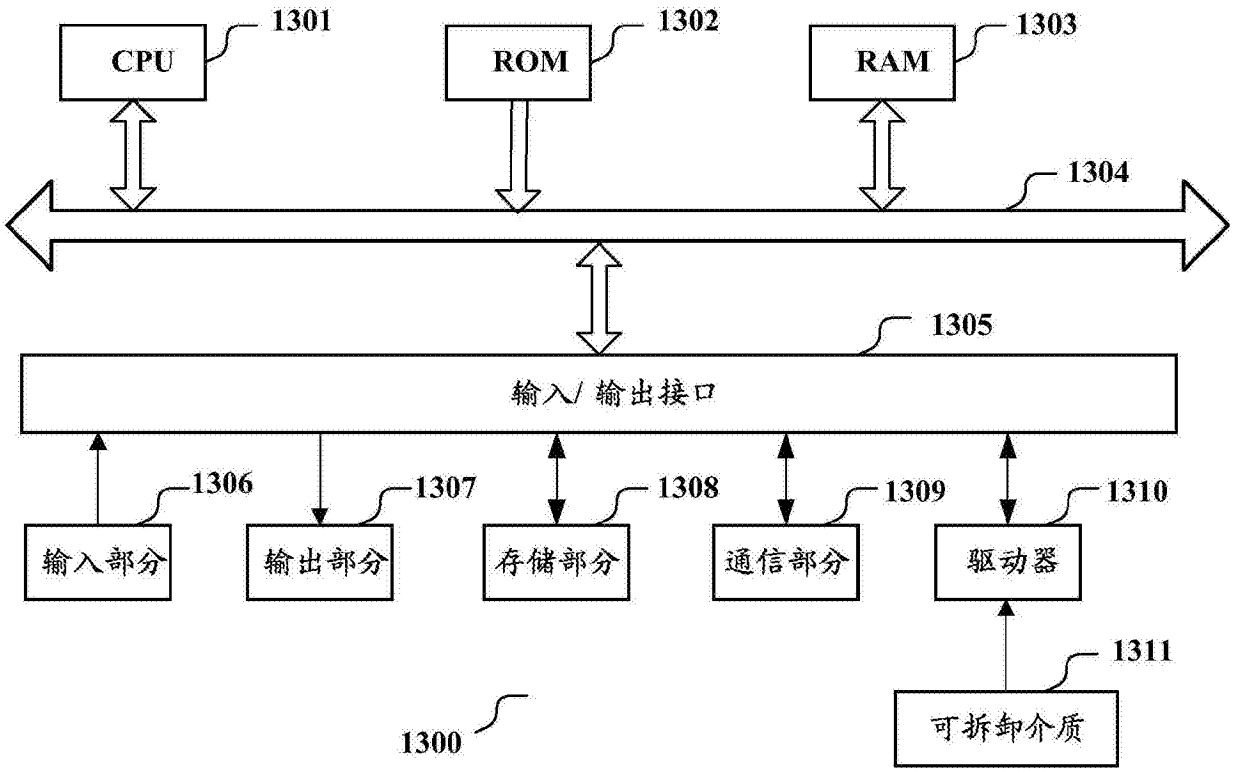

CPU
1301
ROM
1302
RAM
1303
1304
1305
输入/ 输出接口
1306
1307
1308
1309
1310
输入部分
输出部分
存储部分
通信部分
驱动器
1311
可拆卸介质
1300

图13